Vishnu Institute of Technology (Autonomous)
Department of Computer Science and Engineering

Prerequisites for ADSAA Lab

While designing the programs, ensure that the logic is not incorporated in the main method. The main method should function solely as a driver method.

1. Design Java Programs to implement the following Sorting techniques.
    a. Bubble Sort
    b. Selection Sort
    c. Insertion Sort
2. Design Java Programs to implement the following Searching methods.
    a. Linear Search
    b. Binary Search
3. Modify the above programs to calculate and display the time taken to execute the sorting algorithms.
4. Design a Java program to copy the contents of one file into another file.
5. Design a Java program that implements a stack data structure using a custom Stack class.
6. Design a Java program that implements a queue data structure using a custom Queue class.
7. Design a Java program that implements a binary search tree data structure using a custom Binary Search Tree class.

Note:

- The code should be properly indented
- Follow the sample programs shared.

Bubble Sort

```
import java.util.Scanner;

public class BubbleSort {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the number of elements: ");
        int n = sc.nextInt();

        int[] A = new int[n];

        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++)
            A[i] = sc.nextInt();

        bubbleSort(A);

        System.out.println("Sorted array:");
        for (int i = 0; i < A.length; i++)
            System.out.print(A[i] + " ");
    }

    static void bubbleSort(int[] A) {
        int n = A.length;
        for (int i = 0; i <= (n – 2); i++)
            for (int j = 0; j <= (n - 2 – i); j++)
                if (A[j] > A[j + 1]) {
                    // Swap A[j] and A[j + 1]
                    int temp = A[j];
                    A[j] = A[j + 1];
                    A[j + 1] = temp;
                }
    }  //end of method

} //end of class
```

Linear Search

```
import java.util.Scanner;

class LinearSearch {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter size of the array : ");
        int n = sc.nextInt();
        int[] A = new int[n];

        System.out.println("Enter "+ n + " numbers : ");
        for (int i=0; i<n; i++)
            A[i] = sc.nextInt();

        System.out.print("Enter element to search : ");
        int key = sc.nextInt();
        int x = linearsearch(A, key);

        if (x!= -1)
            System.out.println("Element found at location " + x);
        else
            System.out.println("Element not found");
    }

    public static int linearsearch(int[] A, int key){
        for (int i=0; i<A.length; i++)
            if (A[i] == key)
                return i;

        return -1;
    }

}
```

Implementation of Stacks

```
class Stack {
    private int[] stackArray;
    private int top; private
    int capacity;

    // Constructor to initialize the stack
    public Stack(int size) {
        stackArray = new int[size];
        capacity = size;
        top = -1;
    }

    // Method to add an element to the stack
    public void push(int value) {
        if (isFull()) {
            System.out.println("Stack Overflow! Cannot push " + value);
            return;
        }
        stackArray[++top] = value;
        System.out.println(value + " pushed into the stack.");
     }

    // Method to remove and return the top element from the stack
    public int pop() {
        if (isEmpty()) {
            System.out.println("Stack Underflow! Nothing to pop.");
            return -1;
        }
        return stackArray[top--];
    }

    // Method to return the top element without removing it
    public int peek() {
        if (isEmpty()) {
            System.out.println("Stack is empty. Nothing to peek.");
            return -1;
        }
        return stackArray[top];
    }

    // Method to check if the stack is empty
    public boolean isEmpty() {
        return top == -1;
    }

    // Method to check if the stack is full
    public boolean isFull() {
        return top == capacity - 1;
    }

    // Method to print all elements of the stack
    public void printStack() {
        if (isEmpty()) {
            System.out.println("Stack is empty.");
            return;
        }
        System.out.println("Stack elements:");
        for (int i = top; i >= 0; i--) {
            System.out.println(stackArray[i]);
        }
    }

    // Main method to test the Stack class
    public static void main(String[] args) {
        Stack stack = new Stack(5); // Create a stack with capacity of 5

        stack.push(10);
        stack.push(20);
        stack.push(30);
        stack.printStack();

        System.out.println("Top element is: " + stack.peek());
        System.out.println("Popped element is: " + stack.pop());
        stack.printStack();

        System.out.println("Is stack empty? " + stack.isEmpty());
    }
}
```